

**School of Electrical and Electronic Engineering**

**Academic Year 2022/23**

**Semester 1**

**EE4483 Artificial Intelligence and Data Mining**

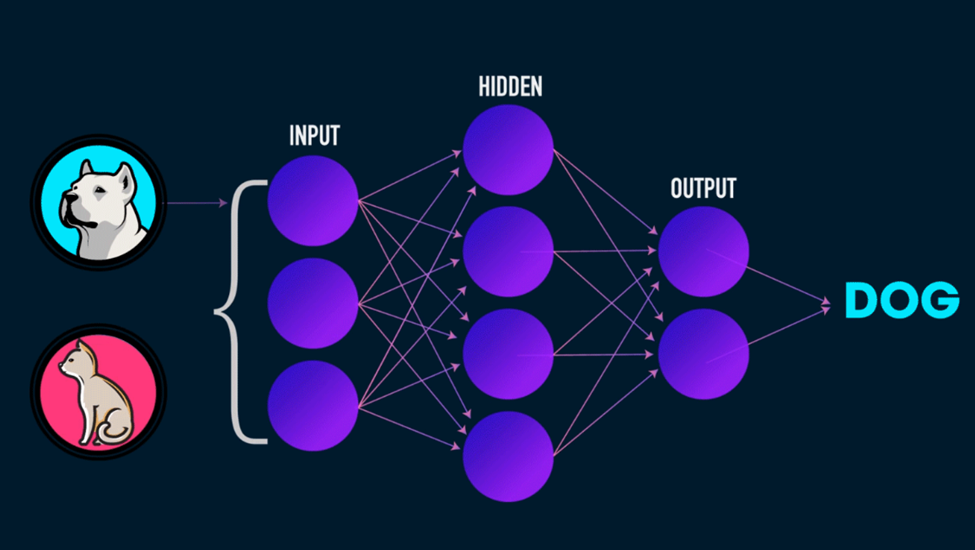
**Mini Project (Option 2)**

Name: WuXiang Jie Kang(U2023490D)

Name: Meng Xu(U2021843B)

Name: Tan Wei En (U2022325F)

**Data Processing Procedures:**



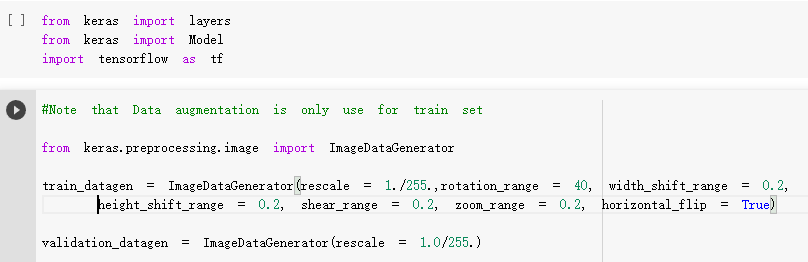
*Figure 1*

The number of images used for training is 2000 and for testing, the set is 500.

For image processing, we imported ImageDataGenerator from the Keras preprocessing module for image processing. There are numerous neural network layers included in Keras, including training-required convolution layers. Additionally, there are layers that don't require any training parameters, like flattened layers that turn an array like an image into a vector.

The technique of data augmentation expands the small training dataset by adding additional images which are variations of available images in the dataset. Training a model with a large dataset (containing available + transformed images) will improve the performance and ability of the AI model to generalize.

For usage in the initial phases of a neural network, the preprocessing layers of the Keras framework were created specifically. It can be used for picture preprocessing tasks such as image scaling, rotation, and brightness/contrast adjustments.

As seen from our code in Figure 2:

*Figure 2*

Rescale= 1/255

The images are rescaled to 1/255 for the efficiency of the algorithms. The reason why we divide it by 255 is to normalize the image to a number from 0 to 1. Image has 3 channels (R, G, B) and each value in the channel can range from 0 to 255. Hence to normalize in the 0 to 1 range, we need to divide it by 255.



*Figure 3*

rotation\_range= 40

The images will be randomly rotated by 40 degrees. One common data augmentation technique is random rotation. A source image is randomly rotated clockwise or counterclockwise by some number of degrees, changing the position of the object in the frame. Notably, for object detection problems, the bounding box must also be updated to encompass the resulting object.

width\_shift\_range=0.2, height\_shift\_range=0.2

Using a floating 0.2 which specifies the upper bound of the fraction of the total width by which the image is to be randomly shifted, either towards the left or right. Hight is exactly like width shifting, except that the image is shifted vertically instead of horizontally. These two arguments are used to generate random data, therefore, making the model more robust.

shear\_range=0.2, zoom\_range=0.2

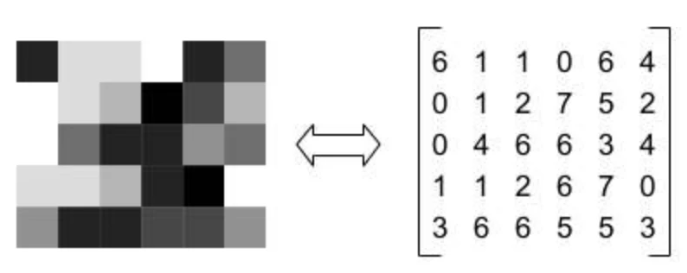
shear\_range=0.2 specifies the angle of the slant in degrees, this creates a sort of ‘stretch’ in the image, which is not seen in rotation. zoom\_range=0.2 magnifies the image.



*Figure 4*

Horizontal\_flip=True

The horizontal\_flip is set to True, the images will be randomly flipped horizontally. Horizontal Flip is a data augmentation technique that takes both rows and columns of such a matrix and flips them horizontally. As a result, you will get an image flipped horizontally along the y-axis.



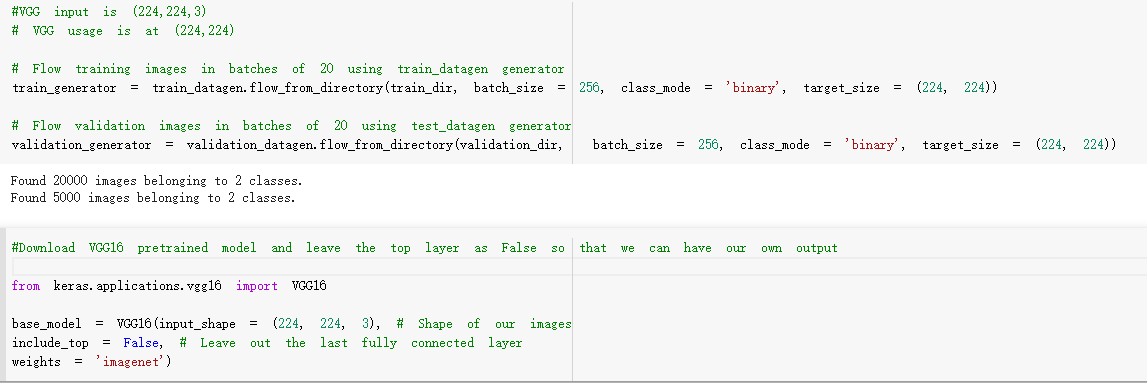
*Figure 5*

**Machine Learning Model Used: VGG16**



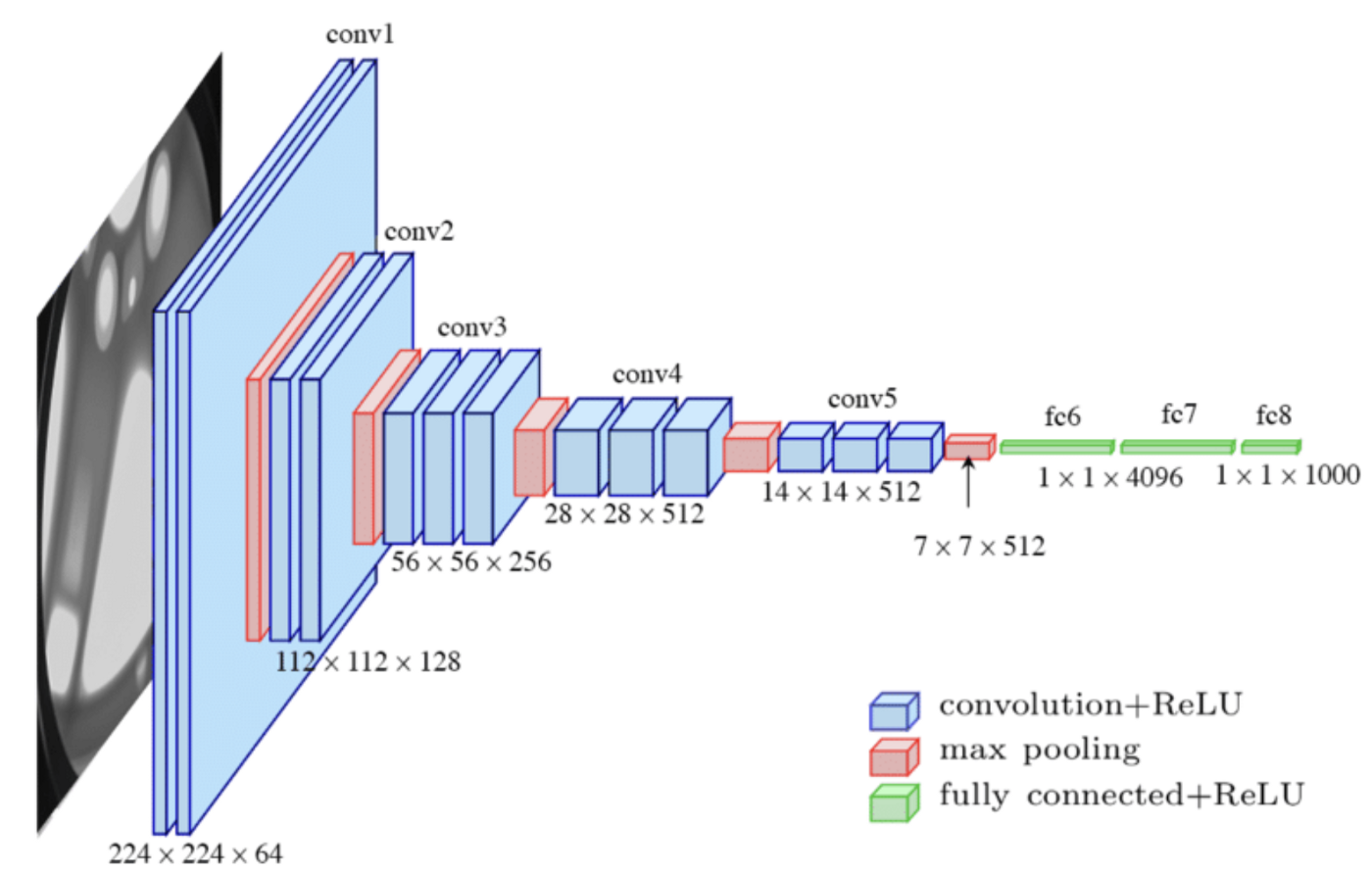
*Figure 6*

For our machine learning model, we are using VGG16 as seen in figure 6 and 7. VGG16 is a type of Convolutional Neural Network (CNN) that is considered to be one of the best computer vision models to date. VGG16 is an object detection and classification algorithm which is able to classify 1000 images of 1000 different categories with 92.7% accuracy. It is one of the popular algorithms for image classification and is easy to use with transfer learning.

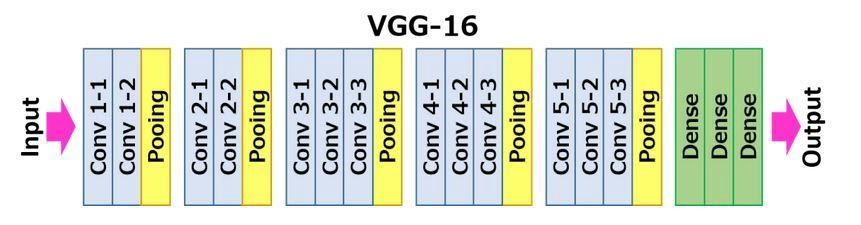


*Figure 7*

**VGG16 Model Architecture**



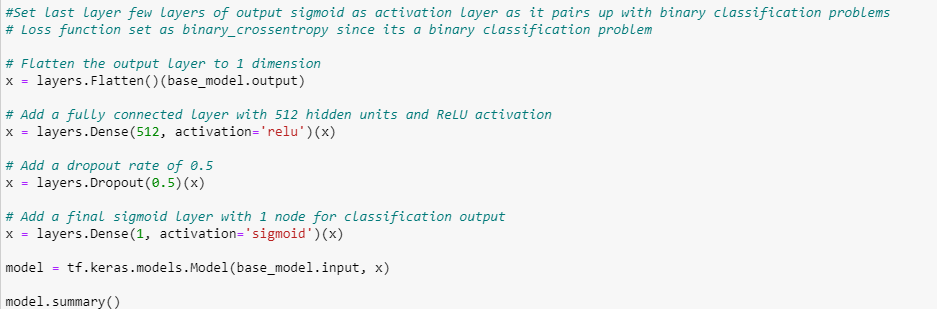
*Figure 8*



*Figure 9*

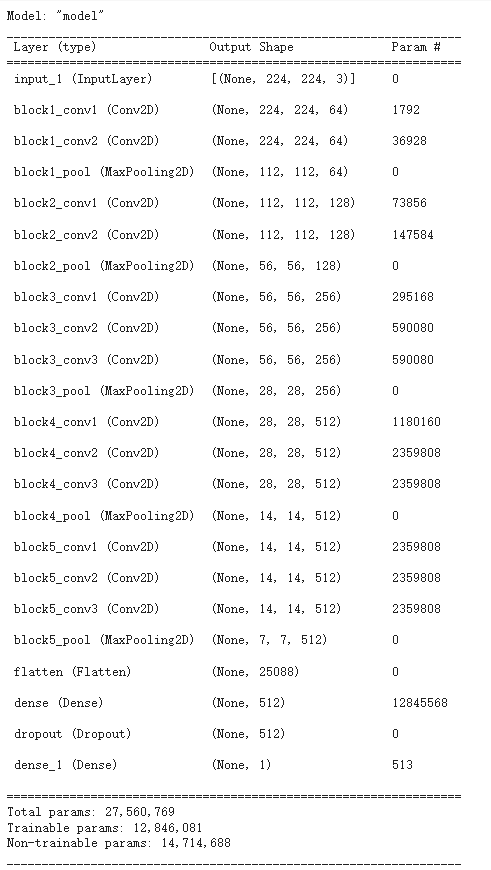
The 16 in VGG16 refers to 16 layers that have weights. In VGG16, there are thirteen convolutional layers, five Max Pooling layers, and three Dense layers which sum up to 21 layers but it has only sixteen weight layers i.e., the learnable parameters layer.

We can check the summary of the model which we created by using the code below in figure 10.



*Figure 10*

The output of this can be seen in the summary of the model in figure 11:



*Figure 11*

The total parameters in the model are 27MB and the size of the model is 769MB.

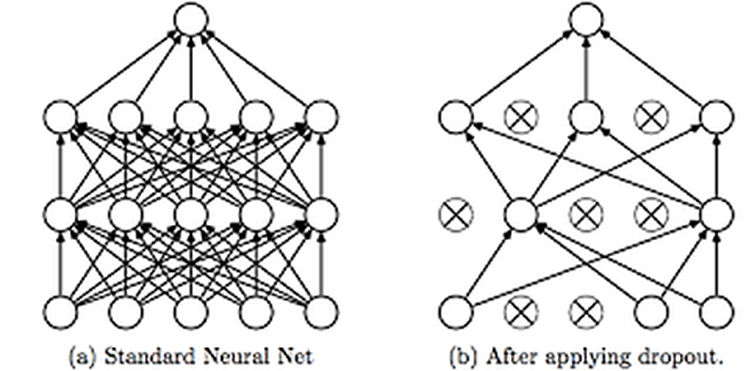
As the number of filters increases following the model depth, hence the number of parameters increases significantly in the later layers. Especially, the parameter number in the two fully connected hidden layers is very large, with total parameters being 27, 560, 769, and trainable parameters being 12, 846, and 081, respectively. It accounts for 86.4% parameters of the whole model.

A large number of parameters may reduce the model performance. Sometimes, it leads to overfitting.

**Overfitting**

Overfitting is a modeling error in statistics that occurs when a function is too closely aligned to a limited set of data points. As a result, the model is useful in reference only to its initial data set, and not to any other data sets.

To solve for overfitting, we set the dropout to 0.5. Dropout is a regularization technique that prevents neural networks from overfitting. Regularization methods like L1 and L2 reduce overfitting by modifying the cost function. Dropout, on the other hand, modifies the network itself. It randomly drops neurons from the neural network during training in each iteration. When we drop different sets of neurons, it’s equivalent to training different neural networks. The different networks will overfit in different ways, so the net effect of dropout will be to reduce overfitting.



*Figure 12*

This technique is shown in figure 12. As we can see, dropouts are used to randomly remove neurons while training the neural network. This technique has proven to reduce overfitting to a variety of problems involving image classification, image segmentation, word embeddings, semantic matching etcetera.

The activation function we used is”relu”. The rectified linear activation function or ReLU for short is a piecewise linear function that will output the input directly if it is positive, otherwise, it will output zero. It has become the default activation function for many types of neural networks because a model that uses it is easier to train and often achieves better performance. We also add a final sigmoid layer with 1 node for classification output, since this is a binary classification problem.

**Loss Function**



*Figure 13*

We used the binary cross entropy loss function when compiling the VGG-16 modal, the loss function tells how good the model is in prediction. Binary cross entropy compares each of the predicted probabilities to the actual class output which can be either 0 or 1. It then calculates the score that penalizes the probabilities based on the distance from the expected value. That means how close or far from the actual value. And this is very suitable for our cats and dog binary classification problem.

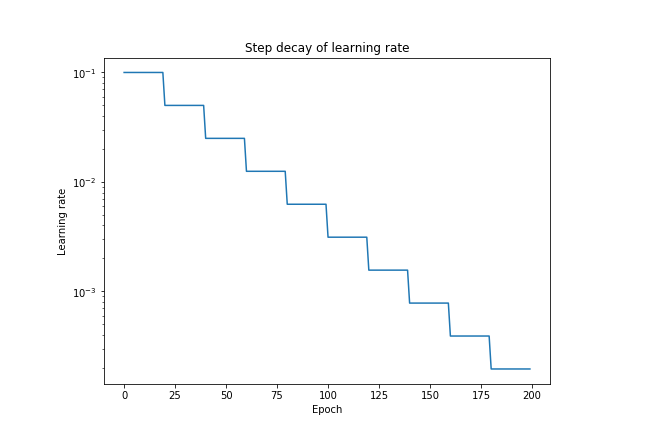
**Feature losses**

The VGG model. a network pre-trained on ImageNet, is used to evaluate the generator model’s loss. The Feature map has 256 channels by 28 by 28 which are used to detect features such as fur, an eyeball, wings, and the type material among many other types of features. The activations at the same layer for the (target) original image and the generated image are compared using mean squared error or the least absolute error (L1) error for the base loss. These are feature losses. This error function uses L1 error.

This allows the loss function to know what features are in the target ground truth image and to evaluate how well the model’s prediction’s features match these rather than only comparing pixel differences. This allows the model being trained with this loss function to produce much finer detail in the generated/predicted features and output.

**Learning Rate**

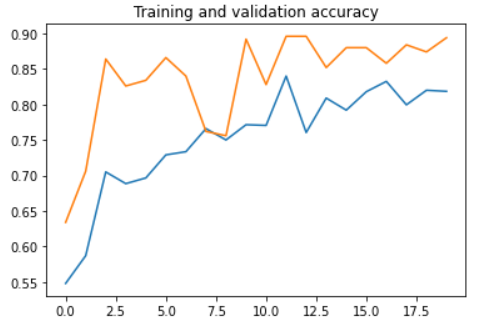
Deep learning models are typically trained by a stochastic gradient descent optimizer. The amount that the weights are updated during training is referred to as the step size or the “learning rate.” The learning rate controls how quickly the model is adapted to the problem. Smaller learning rates require more [training epochs](https://machinelearningmastery.com/difference-between-a-batch-and-an-epoch/) given the smaller changes made to the weights of each update, whereas larger learning rates result in rapid changes and require fewer training epochs. A learning rate that is too large can cause the model to converge too quickly to a suboptimal solution, whereas a learning rate that is too small can cause the process to get stuck. The learning rate is set to 0.0001 to obtain an accurate result since when the learning rate is set to 0.1, we only obtained a result of training accuracy of 60%.



*Figure 14*

**Classification Accuracy**

The training and validation accuracy result is shown in figure 15 below (the orange color is the validation accuracy and the blue one is the training accuracy), the highest validation obtained was at the 12 epoch which is 0.8960. The highest training accuracy obtained was 0.8400.

****

*Figure 15*

****

*Figure 16*

**Strengths & Weaknesses of Model:**

From our test sets, these are some of the images that got classified wrongly as dogs, in the test data, more cats more misclassified as dogs as compared to dogs being misclassified into cats. Figures 17, 18, and 19 are the images that have been classified wrongly as dogs. The VGG-16 model is better in terms of recognizing dogs.

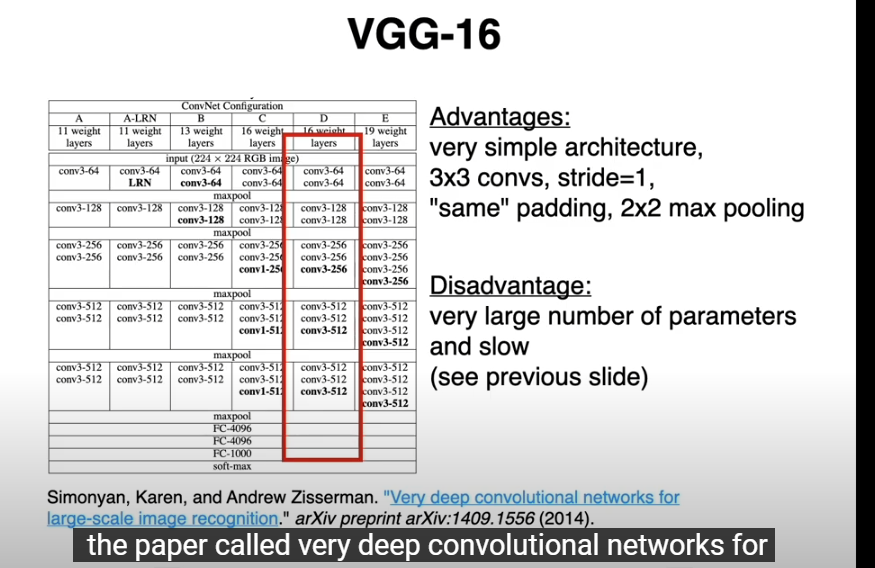
Another interesting fact that we noticed is that most of the misclassified cats shared one similarity. Their fur colors usually contain one light and one dark color and always have some dark color fur on their heads.

****

*Figures 17, 18, 19*

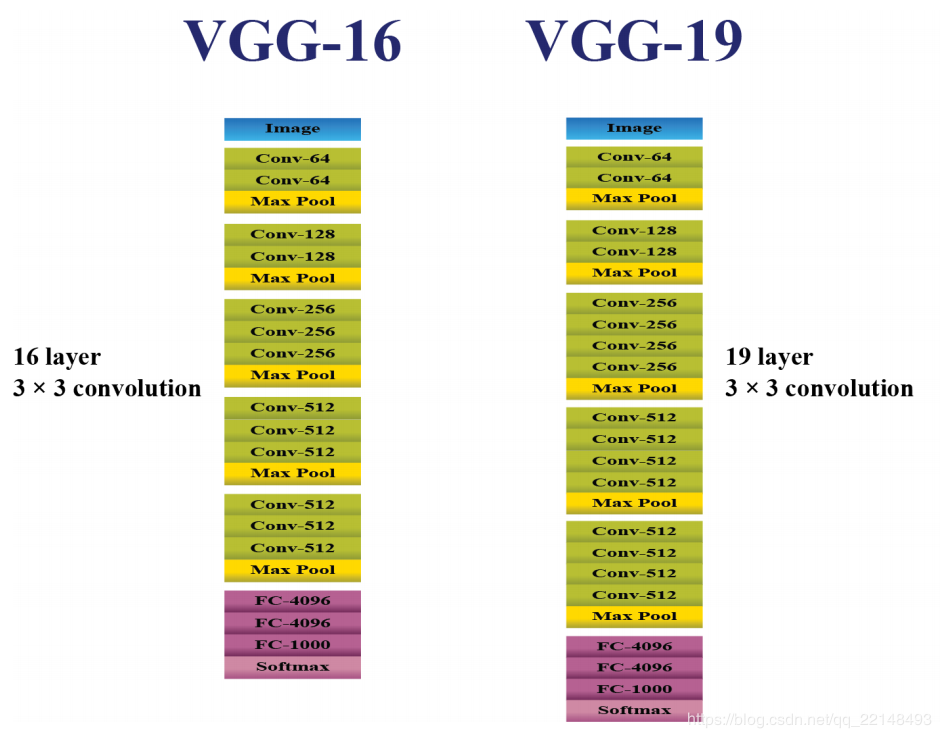
Advantage: TheVGG-16 has a very simple architecture, therefore the modal is quite easy to implement. It is based on just using a 3 by 3 convolution and a 2 by 2 max pooling. Still works well to get a good enough baseline to get a score of close to 89.6% on classical problems like cat vs dog classification.

Disadvantage: One of the crucial downsides of the VGG16 network is that it is a huge network, which means that it takes more time to train its parameters. Due to its depth and number of fully connected layers, the VGG16 model is more than 533MB. This makes implementing a VGG network a time-consuming task.

****

*Figure 20*

**Another Choice of model: VGG19**

****

*Figure 21*

Figure 21 shows the comparison between VGG16 and VGG19.

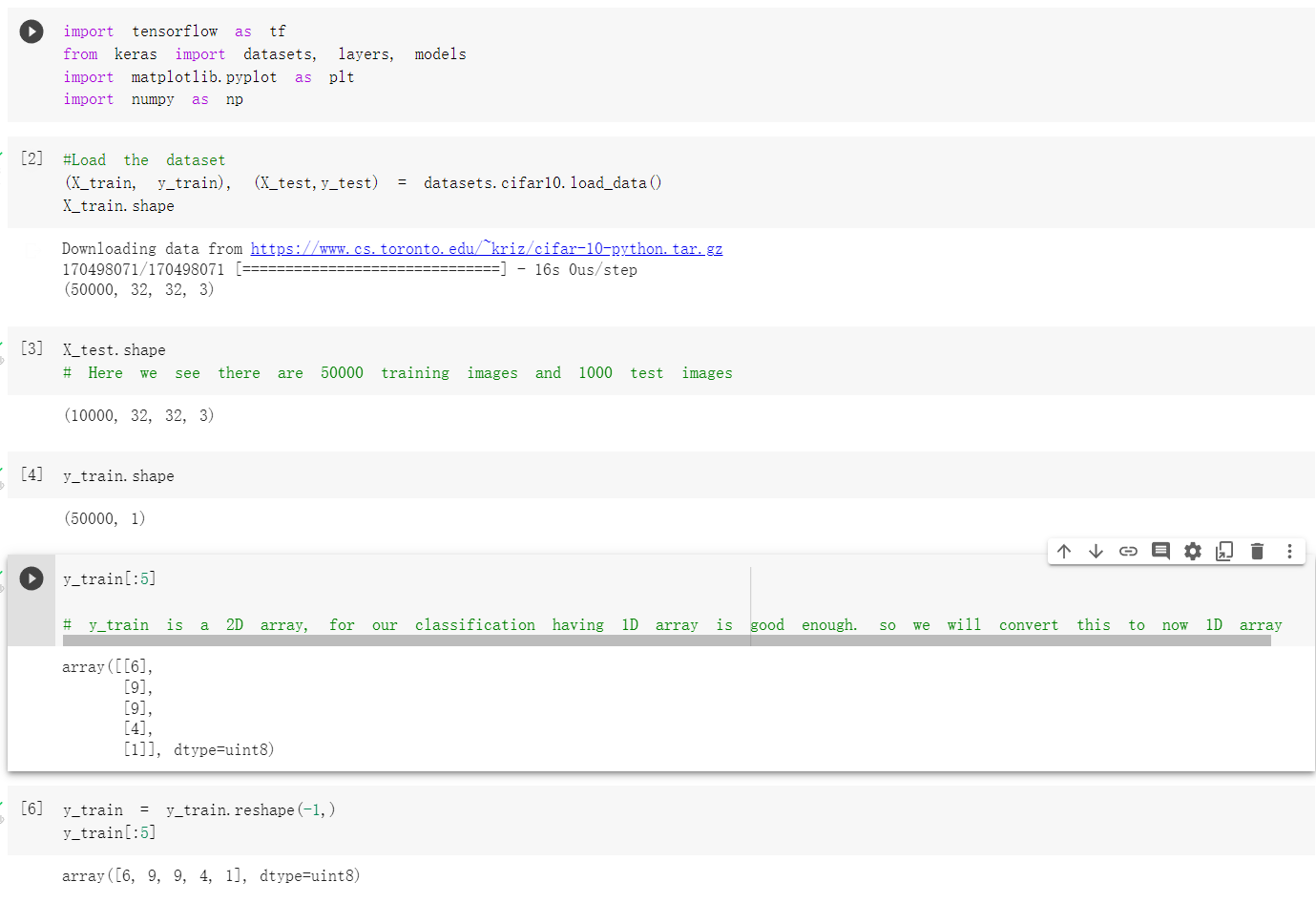
The concept of the VGG19 model (also VGGNet-19) is the same as the VGG16 except that it supports 19 layers. The “16” and “19” stand for the number of weight layers in the model (convolutional layers). This means that VGG19 has three more convolutional layers than VGG16.

With more layers, a problem arises the weights of a neural network are updated through the backpropagation algorithm, which makes a minor change to each weight so that the loss of the model decreases. It updates each weight so that it takes a step in the direction along which the loss decreases. This is nothing but the gradient of this weight which can be found using the chain rule. However, as the gradient keeps flowing backward to the initial layers, the value keeps increasing with each local gradient. This results in the gradient becoming smaller and smaller, thereby making changes to the initial layers very small. This, in turn, increases the training time significantly.

If more training data is generated with image processing, the more accurate will the validation set be.

**Cifar-10 Dataset**

The CIFAR-10 dataset consists of 60000 32x32 color images in 10 classes, with 6000 images per class. There are 50000 training images and 10000 test images. We first reshape the y\_train and y\_test as seen in figure 22, to a 1D array, for our classification 1D array is good enough.



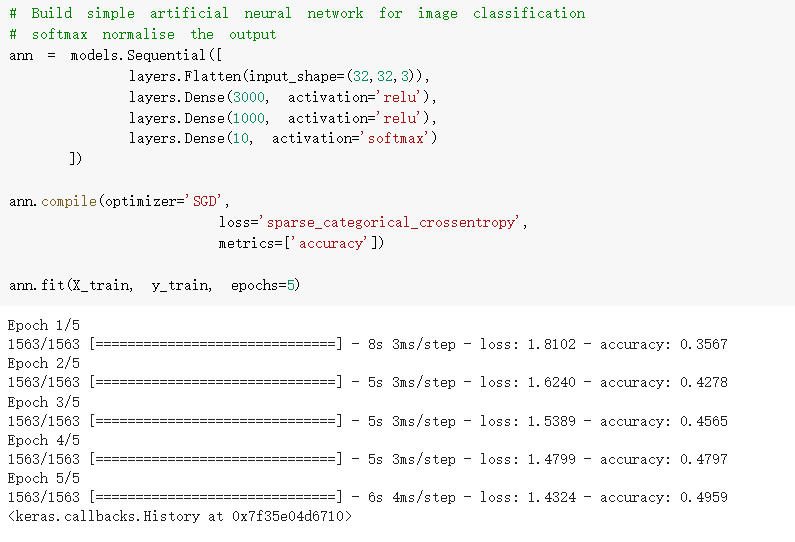
*Figure 22*

We build an array of classes and plot some of the sample data in figure 23, and we also resize the training data size by dividing 255.



*Figure 23*

An artificial neural network is used for training with an input shape of 32\*32, however, the accuracy was only around 49%. So we decide to use a convolutional neural network.



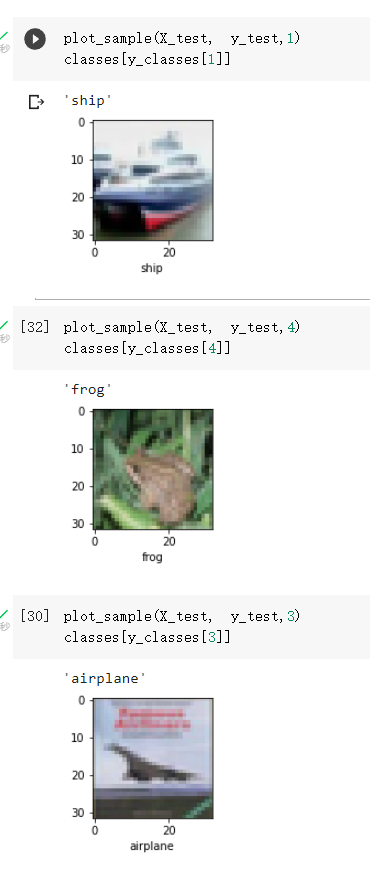
*Figure 24*

After we build the convolutional neural network with 3 by 3 kernel size and Relu for the activation function, we also used the software activation function to generalize the output. For the compilation of the model, we applied a sparse categorical cross-entropy function for the loss function. Following is the definition of cross-entropy when the number of classes is larger than 2 (which is in our case).

Definition: Sparse Categorical Cross-entropy and multi-hot categorical cross-entropy use the same equation and should have the same output. The difference is both variants cover a subset of use cases and the implementation can be different to speed up the calculation.

The train accuracy obtained after 10 epochs is around 94.41% while the test accuracy was around 66.95%, since they are so many classes to be classified, this is considered a decent result.

After building the training of the model, we tried some of the test images and all obtained the correct result as seen in figure 25.



*Figure 25*

**Approach used to improve our algorithm to tackle the data unbalancing issue:**

First Approach: We can change the dataset that we use to build your predictive model to have more balanced data.

This change is to sample the dataset and there are two main methods that we use to even up the classes:

1. We can add copies of instances from the under-represented class called over-sampling (or more formally sampling with replacement), or
2. We can delete instances from the over-represented class, called under-sampling.

These approaches are often very easy to implement and fast to run. They are an excellent starting point. We can try to use SMOTE (**Synthetic Minority Oversampling Technique**) Algorithm to synthesize new examples for the minority class.

Second Approach: Use Keras ImageDataGenerator to perform image augmentation, so that the dataset size is increased.

**References:**

*“Step by step VGG16 implementation in Keras for beginners,” Vgg16 Implementation In Keras, 06-Aug-2019. [Online]. Available:* [*https://towardsdatascience.com/step-by-step-vgg16-implementation-in-keras-for-beginners-a833c686ae6c*](https://towardsdatascience.com/step-by-step-vgg16-implementation-in-keras-for-beginners-a833c686ae6c)*. [Accessed: 09-Nov-2022].*

*G. Learning, “Everything you need to know about VGG16,” Medium, 23-Sep-2021. [Online]. Available:* [*https://medium.com/@mygreatlearning/everything-you-need-to-know-about-vgg16-7315defb5918*](https://medium.com/@mygreatlearning/everything-you-need-to-know-about-vgg16-7315defb5918)*. [Accessed: 09-Nov-2022].*

*SAROJ BHATTARAI, “Cat and dog classifier using VGG16 model,” Kaggle, 16-Sep-2018. [Online]. Available:* [*https://www.kaggle.com/code/thevirusx3/cat-and-dog-classifier-using-vgg16-model*](https://www.kaggle.com/code/thevirusx3/cat-and-dog-classifier-using-vgg16-model)*. [Accessed: 09-Nov-2022].*